PERGAMON Computers and Mathematics with Applications 41 (2001) 1403–1416

An International Journal
computers & mathematics
with applications

www.elsevier.nl/locate/camwa

Systolic Implementation of Real-Valued Discrete Transforms via Algebraic Integer Quantization

R. BAGHAIE AND V. DIMITROV
Signal Processing Laboratory, Helsinki University of Technology
P.O. Box 3000, 02015 HUT, Finland
Ramin.Baghaie@hut.fi
vdimitro@wooster.hut.fi

(Received September 1999; revised and accepted March 2000)

Abstract—In this paper, we propose a novel approach for computing real-valued discrete transforms such as the discrete cosine transform (DCT) and the discrete Hartley transform (DHT). The approach is based on the algebraic integer encoding scheme. With the aid of this scheme, an error-free representation of the *cos*, *sin*, and *cas* functions becomes possible. For further complexity reduction, two different approximation methods are presented. Furthermore, for the implementation of these algorithms, a fully pipelined systolic architecture with $O(N)$ throughput is proposed.

Keywords—Discrete cosine transform, Discrete Hartley transform, Systolic arrays, Exact computer arithmetic, Haar transform.

1. INTRODUCTION

In many digital signal processing (DSP) applications such as speech and image processing, the input data is real-valued. Furthermore, there exists a wide range of real-valued orthogonal transforms dedicated to overcome the need of using the complex discrete Fourier transform (DFT). Among the most popular are the discrete cosine transform (DCT) [1], the discrete Hartley transform (DHT) [2], the discrete sine transform (DST) [3], and the Haar transform [4]. In this paper, we will mainly concentrate on the first two transforms, the DCT and the DHT, and propose a novel approach to compute them via algebraic integer quantization. The proposed method can also be applied to other real-valued orthogonal transforms.

The DCT has an important role in numerous image processing applications such as image compression standards [5]. The 1-D DCT of an N-point sequence $\{x_n,\ n = 0, \ldots, N-1$ and

This work is part of a research project of the Institute of Radio Communication (IRC) funded by the National Technology Agency (TEKES), NOKIA Research Center, Sonera, and Elisa Communications.
The authors would like to thank the anonymous reviewers and Professor T. Laakso at Helsinki University of Technology for the valuable comments and suggestions for improving this paper.

0898-1221/01/$ - see front matter Typeset by AMS-TEX
PII: S0898-1221(01)00105-5

$N = 2^m$, $m \geq 1\}$ is defined as [1]

$$X_k = \frac{2}{N} e(k) \sum_{n=0}^{N-1} x_n \cos\left(\frac{\pi}{2N}(2n+1)m\right), \qquad 0 \leq k \leq N-1, \tag{1}$$

where $e(k) = 1/\sqrt{2}$ if $k = 0$; otherwise $e(k) = 1$. Since the factor $2e(k)/N$ results only in a slight modification of X_k, it is sufficient to consider the following DCT-like equation:

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left(\frac{\pi}{2N}(2n+1)m\right), \qquad 0 \leq k \leq N-1. \tag{2}$$

The DHT is an attractive alternative to the discrete Fourier transform (DFT) because of its real-valued computation and properties similar to those of the DFT [1]. Another interesting property of the DHT is that the same kernel is used for both the transform and its inverse transform. Consequently, since its introduction, the DHT has found its way to many digital signal processing applications [6]. The 1-D DHT of an N-point sequence $\{x_n,\ n = 0, \ldots, N-1$ and $N = 2^m$, $m \geq 1\}$ is defined as [2]

$$X_k = \sum_{n=0}^{N-1} x_n \operatorname{cas}\left(2\pi \frac{kn}{N}\right), \qquad 0 \leq k \leq N-1, \tag{3}$$

where $\operatorname{cas}\theta = \cos\theta + \sin\theta$.

Since the introduction of the DCT and the DHT, a number of systolic architectures have been proposed, many of which are based on the direct implementation of the algorithm [3,7–9]. In these implementations, for the calculation of the *cos* and the *cas* functions, different types of approximations have been introduced. Processing with algebraic integers, in which the signal sample is represented by a set of small integers, was introduced in [10]. Algebraic integers are roots of monic polynomials that have integer coefficients with leading coefficient equal to unity. The motivation for introducing this new mapping of real numbers is to drastically reduce the dynamic range of each of the independent computations.

In this paper, we illustrate how with the aid of the algebraic integer encoding scheme, an efficient error-free implementation of the real-valued discrete transforms is possible. This paper is organized as follows. Section 2 briefly reviews the theory of algebraic integers. Furthermore, algebraic integers interpretation of two real-valued discrete transforms is presented. Finally, in this section, we present a procedure on how to apply the proposed scheme to other real-valued discrete transforms. In Section 3, for further complexity reduction, two different approximation schemes are proposed. In Section 4, for the implementation of the proposed algorithms, a novel systolic array is presented. In Section 5, certain hardware and throughput issues are discussed. Concluding remarks are provided in Section 6.

2. THEORETICAL BACKGROUND

Computation of the DFT requires approximation of the complex roots of unity, and this inevitably leads to computational errors. If integer processing is to be utilized, then a large dynamic range is required even for a modest amount of precision. In [10], Cozzens and Finkelstein introduced a new approach for computing the DFT that uses the residue number system (RNS) in a ring of algebraic integers. Algebraic integers combine with RNS processing to add a second level of parallelism to integer RNS processing generalizing the quadratic residue number system (QRNS) concept. Algebraic integers are roots of monic polynomials that have integer coefficients with the leading coefficient equal to unity.

Let $\omega = e^{2\pi j/16}$ denote the primitive 16^{th} root of unity. Then, ω satisfies the equation $x^8 + 1 = 0$. If ω is adjoined to the rational numbers, then the associated ring of algebraic integers is denoted

by $Z[\omega]$. The ring $Z[\omega]$ can be regarded as consisting of polynomials in ω of degree 7 with integer coefficients. The elements of $Z[\omega]$ are added and multiplied as polynomials, except that the rule $\omega^8 = -1$ is used in the product to reduce the degree of powers of ω to below 8. For an integer M, $Z[\omega]_M$ is used to denote the elements of $Z[\omega]$ with coefficients between $-M/2$ and $M/2$. A real number x in $Z[\omega]$ can be written in the form

$$x = a_0 + \sqrt{2+\sqrt{2}}\,a_1 + \sqrt{2}\,a_2 + \sqrt{2-\sqrt{2}}\,a_3. \tag{4}$$

The ring of all such elements is denoted by $Z[\sqrt{2+\sqrt{2}}]$. If $\theta = \sqrt{2+\sqrt{2}}$, then θ is a root of the polynomial $x^4 - 4x^2 + 2 = 0$ and the elements of $Z[\theta]$ have a polynomial form, where the relation $\theta^4 = 4\theta^2 - 2$ is used to reduce powers of θ above three.

The elements of $Z[\theta]$ are used to process separately the real and imaginary part of $Z[\omega]$. In summary, algebraic integers of an extension of degree n are of the form

$$a_0\omega_0 + a_1\omega_1 + \cdots + a_{n-1}\omega_{n-1}, \tag{5}$$

where $\{\omega_0, \omega_1, \ldots, \omega_{n-1}\}$ is called the algebraic-integer *basis* and the coefficients a_i are integers.

Algebraic-Integer Interpretation of DCT and DHT

Let us consider the eight-point DCT. The kernel of this transformation is $\cos((\pi/2N)(2n+1)m)$ where $0 \le k, n \le N-1$. The classical method for calculating the *cos* function is to approximate it in a binary number system, which leads to round-off errors. In this paper, we adopt the algebraic integer encoding scheme. Consider the first nonzero angle of the *cos* function that is $\pi/16$. We can represent the *cos* function of this angle as [11]

$$\cos\left(\frac{\pi}{16}\right) = \frac{1}{2}\sqrt{2+\sqrt{2+\sqrt{2}}}. \tag{6}$$

The other needed angles can be represented in a similar manner as follows:

$$\cos\left(\frac{2\pi}{16}\right) = \frac{1}{2}\sqrt{2+\sqrt{2}}, \tag{7}$$

$$\cos\left(\frac{3\pi}{16}\right) = \frac{1}{2}\sqrt{2+\sqrt{2-\sqrt{2}}}, \tag{8}$$

$$\cos\left(\frac{4\pi}{16}\right) = \frac{\sqrt{2}}{2}, \tag{9}$$

$$\cos\left(\frac{5\pi}{16}\right) = \frac{1}{2}\sqrt{2-\sqrt{2-\sqrt{2}}}, \tag{10}$$

$$\cos\left(\frac{6\pi}{16}\right) = \frac{1}{2}\sqrt{2-\sqrt{2}}, \tag{11}$$

$$\cos\left(\frac{7\pi}{16}\right) = \frac{1}{2}\sqrt{2-\sqrt{2+\sqrt{2}}}. \tag{12}$$

Without compromising the calculations, we omit the "2" from the denominator of (6). Let us denote z_1 as $z_1 = 2\cos(\pi/16) = \sqrt{2+\sqrt{2+\sqrt{2}}}$, where z_1 is a root of the following equation [11]:

$$x^8 - 8x^6 + 20x^4 - 16x^2 + 2 = 0. \tag{13}$$

Consider the following polynomial expansion:

$$f(z_1) = \sum_{i=0}^{7} a_i z_1^i, \tag{14}$$

Table 1. Representation of the *cos* function for the eight-point DCT [11].

	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	Error
$2\cos(0\pi/16)$	2	0	0	0	0	0	0	0	0
$2\cos(1\pi/16)$	0	1	0	0	0	0	0	0	0
$2\cos(2\pi/16)$	−2	0	1	0	0	0	0	0	0
$2\cos(3\pi/16)$	0	−3	0	1	0	0	0	0	0
$2\cos(4\pi/16)$	2	0	−4	0	1	0	0	0	0
$2\cos(5\pi/16)$	0	5	0	−5	0	1	0	0	0
$2\cos(6\pi/16)$	−2	0	9	0	−6	0	1	0	0
$2\cos(7\pi/16)$	0	−7	0	14	0	−7	0	1	0

where a_i $(i = 0, \ldots, 7)$ are integers. By assigning $(0, 1, 0, 0, 0, 0, 0, 0)$ to a_i, we have an *exact code* for z_1, and thus for $2\cos(\pi/16)$ [11]. Proceeding in the same manner, we can obtain an *error-free* representation of the *cos* function necessary to evaluate the eight-point 1-D DCT [11]. Table 1 presents the corresponding coefficients of every required angle.

Other remaining angles can be obtained by changing the signs of the given coefficients. This means that the multiplication of integer number x times $2\cos(4\pi/16)$ can now be replaced by vector $(2x, 0, -4x, 0, x, 0, 0, 0)$. Therefore, the aforementioned vector is the error-free representation of $x \cdot 2\cos(4\pi/16)$. This means that multiplication by the integers in Table 1 can now be replaced by maximum two shift operations and one addition.

Now, consider the 16-point DHT. The kernel of this transformation is $\mathrm{cas}(2kn\pi/16)$ where $0 \leq k, n \leq N - 1$. Consider the first nonzero angle of the *cas* function that is $2\pi/16$. We can represent the *cos* and *sin* functions of this angle as

$$\cos\left(\frac{2\pi}{16}\right) = \frac{1}{2}\sqrt{2+\sqrt{2}}, \tag{15}$$

$$\sin\left(\frac{2\pi}{16}\right) = \frac{1}{2}\sqrt{2-\sqrt{2}}. \tag{16}$$

The other needed angles can be represented in a similar manner. Denote z_2 as $z_2 = 2\cos(2\pi/16) = \sqrt{2+\sqrt{2}}$, where z_2 is a root of the following equation:

$$x^4 - 4x^2 + 2 = 0. \tag{17}$$

Consider now the following polynomial expansion:

$$f(z_2) = \sum_{i=0}^{3} a_i z_2^i. \tag{18}$$

By assigning $(0, 1, 0, 0)$ to a_i, we have an *exact code* for z_2. Similarly, by assigning $(0, -3, 0, 1)$ to a_i, we have an exact code for $2(\sin 2\pi/16)$. Consequently, $(0, -2, 0, 1)$ is an exact representation of $2(\mathrm{cas}\, 2\pi/16)$. Proceeding in the same manner as in the DCT case, we can obtain an *error-free* representation of the *cas* function necessary to evaluate the 16-point DHT. Table 2 presents the corresponding coefficients of every required angle. Other remaining angles can be obtained by changing the signs of the given coefficients. It is worth mentioning that in this case, multiplication is replaced with *only* one shift operation [12].

Up to this point, for the computation of the DCT and the DHT, an error-free presentation has been utilized. However, the accuracy of the final reconstruction depends on the precision

Table 2. Representation of the *cas* function for the 16-point DHT.

	a_0	a_1	a_2	a_3	Error
$2\cos(0\pi/16)$	2	0	0	0	0
$2\cos(2\pi/16)$	0	−2	0	1	0
$2\cos(4\pi/16)$	−4	0	2	0	0
$2\cos(6\pi/16)$	0	−2	0	1	0

used to represent z_1 or z_2. Therefore, one can estimate the precision that is needed to ensure the required accuracy. As an example, if the word-length of the input data stream is eight-bit, then $\hat{z}_1 = 2 - 2^{-5} - 2^{-7}$ is a very good approximation of $z_1 = 1.961570561\ldots$ with 10 bits of accuracy.

For the final reconstruction, in order to reduce the computational complexity of the polynomials of (14) and (18), Horner's rule is utilized [13]. There are, however, alternative methods to Horner's rule. The interested reader may refer to [13]. We have applied Horner's rule due to its simplicity and straightforward applicability to systolic array. Equation (19) illustrates the application of Horner's rule to (18). The same method can also be applied to (14) for the DCT case

$$f(z_2) = ((a_3 z_2 + a_2)z_2 + a_1)z_2 + a_0. \tag{19}$$

Application to Other Transforms

The proposed method can be applied to other real-valued orthogonal transforms such as the DST or the Haar transform. The procedure of computing other transforms is the following.

The first step in applying the proposed idea to other transforms is to identify the appropriate algebraic integer encoding scheme. The kernel of the encoding scheme has to be chosen as an algebraic integer of the highest possible degree, which is used in the transformation matrix. Once the encoding scheme is identified, the second step is to obtain the error-free representation of the transformation matrix. With this representation in hand, one can directly apply the above implementation approach.

An example for applying the same idea to other transforms is the computation of the second-order Haar transform. In this case, the transformation matrix consists of the numbers 0, ± 1, and $\pm\sqrt{2}$ (see [14]). Since $\sqrt{2}$ is an algebraic integer of second degree, it is quite convenient to choose algebraic numbers of the form $a_0 + \sqrt{2}a_1$, where a_0 and a_1 are integers. Now, we only have to work with a pair of small integers (a_0, a_1). Table 3 represents the corresponding coefficients for the algebraic integer encoding scheme of the second-order Haar transform. Therefore, the use of algebraic integer quantization leads to error-free implementation and allows us to avoid rounding errors. The final reconstruction of the result uses an appropriate approximation of $\sqrt{2}$ in accordance to the required precision.

Table 3. Representation of the transformation matrix elements of the second-order Haar transform.

	a_0	a_1	Error
0	0	0	0
+1	1	0	0
−1	−1	0	0
$+\sqrt{2}$	0	1	0
$-\sqrt{2}$	0	−1	0

Table 4. Approximation of $\cos(2\pi k/32)$ for the 16-point DCT with no constraint.

k	Four-Bit Dynamic Range					Five-Bit Dynamic Range					Six-Bit Dynamic Range				
	a_0	a_1	a_2	a_3	Error	a_0	a_1	a_2	a_3	Error	a_0	a_1	a_2	a_3	Error
0	2	0	0	0	0	2	0	0	0	0	2	0	0	0	0
1	−8	−2	−4	0	0.0002344414	9	8	1	−4	0.0001392826	−24	31	13	−12	0.0000080482
2	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0
3	−3	−8	2	2	0.0007034996	−1	1	−9	5	0.0001180780	8	4	20	−13	0.0000044944
4	−2	0	1	0	0	−2	0	1	0	0	−2	0	1	0	0
5	−6	2	1	0	0.0014087736	7	−16	−6	7	0.0000584621	−9	7	−23	12	0.0000102279
6	0	−3	0	1	0	0	−3	0	1	0	0	−3	0	1	0
7	4	6	−8	2	0.0000466333	4	6	−8	2	0.0000466333	−12	29	−25	7	0.0000016009
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5. Approximation of $\mathrm{cas}(2\pi k/32)$ for the 32-point DHT with no constraint.

k	Four-Bit Dynamic Range					Five-Bit Dynamic Range					Six-Bit Dynamic Range				
	a_0	a_1	a_2	a_3	Error	a_0	a_1	a_2	a_3	Error	a_0	a_1	a_2	a_3	Error
0, 8	2	0	0	0	0	2	0	0	0	0	2	0	0	0	0
1, 7	−4	4	−4	2	0.0002810746	3	−12	10	−2	0.0000124810	3	−12	10	2	0.0000124810
2, 6	0	−2	0	1	0	0	−2	0	1	0	0	−2	0	1	0
3, 5	−7	3	−8	5	0.0012906956	−1	11	−3	−1	0.0000157223	−20	−18	−15	17	0.0000027199
4	−4	0	2	0	0	−4	0	2	0	0	−4	0	2	0	0
9, 15	−1	−6	4	0	0.0010900574	14	−5	12	−7	0.0001309120	5	−8	−17	11	0.0000083340
10, 14	0	−4	0	1	0	0	−4	0	1	0	0	−4	0	1	0
11, 13	−7	−7	6	0	0.0008308395	8	−16	12	10	0.0000659494	−25	17	26	−15	0.0000029285
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

3. APPROXIMATION SCHEMES

In the previous section, we illustrated that for an error-free presentation of the *cos* function that is required in the computation of the N-point DCT, I the degree of the polynomial of (14) is of length $N-1$. By taking into account different existing DCT algorithms, this degree can be reduced.

In [11], by applying the Feig and Winograd algorithm [15], I was reduced to $N/2-1$. It is important to note that in order to further reduce the degree of the polynomial, the proposed approach can be combined with *any* other DCT algorithm. In the case of the N-point DHT, for an error-free calculation of *cas* function I is $N/4-1$. Similarly, we can apply this approach to any DHT algorithm to further reduce the computational complexity.

Nevertheless, for an error-free presentation of the *cos* or *cas* functions, the length of a_i $(0 \le i \le I)$ and correspondingly the sizes of Tables 1 and 2 are linearly dependent on N. Therefore, for a large N, I is relatively large. Thus, direct implementation of these tables may not be practical in an application. However, by employing different approximation techniques, tremendous hardware reductions can be achieved, and yet a very good estimate of the aforementioned functions can be obtained. In these approximations, we assign a fixed length to I that is independent of N. As a

Table 6. Approximation of $\cos(2\pi k/32)$ for the 16-point DCT with the $(\pm 2^p \pm 2^q)$ constraint.

k	Four-Bit Dynamic Range					Five-Bit Dynamic Range					Six-Bit Dynamic Range				
	a_0	a_1	a_2	a_3	Error	a_0	a_1	a_2	a_3	Error	a_0	a_1	a_2	a_3	Error
0	2	0	0	0	0	2	0	0	0	0	2	0	0	0	0
1	−8	−2	−4	0	0.0002344414	9	8	1	−4	0.0001392826	−1	−18	18	4	0.0000341523
2	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0
3	−3	−8	2	2	0.0007034996	−1	1	−9	5	0.0001180780	15	−32	−18	17	0.0001244115
4	−2	0	1	0	0	−2	0	1	0	0	−2	0	1	0	0
5	−6	2	1	0	0.0014087736	7	−16	−6	7	0.0000584621	7	−16	−6	7	0.0000584621
6	0	−3	0	1	0	0	−3	0	1	0	0	−3	0	1	0
7	4	6	−8	2	0.0000466333	4	6	−8	2	0.0000466333	30	9	−8	−3	0.0000102628
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 7. Approximation of $\mathrm{cas}(2\pi k/32)$ for the 32-point DHT with the $(\pm 2^p \pm 2^q)$ constraint.

k	Four-Bit Dynamic Range					Five-Bit Dynamic Range					Six-Bit Dynamic Range				
	a_0	a_1	a_2	a_3	Error	a_0	a_1	a_2	a_3	Error	a_0	a_1	a_2	a_3	Error
0, 8	2	0	0	0	0	2	0	0	0	0	2	0	0	0	0
1, 7	−4	4	−4	2	0.0002810746	3	−12	10	2	0.0000124810	3	−12	10	−2	0.0000124810
2, 6	0	−2	0	1	0	0	−2	0	1	0	0	−2	0	1	0
3, 5	−7	3	−8	5	0.0012906956	6	−15	−15	12	0.0000596160	−20	−18	−15	17	0.0000027199
4	−4	0	2	0	0	−4	0	2	0	0	−4	0	2	0	0
9, 15	−1	−6	4	0	0.0010900574	14	−5	12	−7	0.0001309120	31	−5	−17	6	0.0000485566
10, 14	0	−4	0	1	0	0	−4	0	1	0	0	−4	0	1	0
11, 13	−7	−7	6	0	0.0008308395	8	−16	−12	10	0.0000659494	−2	−32	31	−7	0.0000231757
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

result, the degree of the polynomial only depends on the dynamic range of the input data that accordingly dictates the accuracy of the computations.

Let us demonstrate this by means of an example. Tables 4 and 5 illustrate the case for the 16-point DCT and the 32-point DHT, respectively, in which I is fixed and equal to three. Consequently, some error is introduced. As an example, vector $(4, 6, -8, 2)$ is a good approximation of $\cos(14\pi/32)$ with an error of 0.0000456333 (see Table 4). This means that by having $I = 3$, and assigning *only* five bits to the dynamic range of the coefficients, 14-bit accuracy is achieved.

By observing the integers in Tables 4 and 5, one can see that in many cases, more than two shift operations maybe required. One of the most attractive ways to obtain efficient multiplierless structure in the field of DSP is to design the coefficients used as a sum of a very small number of powers of two. That is why by applying the constraint $\pm 2^p \pm 2^q$, where p and q are integers within the specific dynamic range, we can reduce the computation cost of each multiplication to maximum two shifts and one addition (see Tables 6 and 7). Let us demonstrate the significance of the aforementioned constraint by means of an example. Consider the 16-point DCT when

Table 8. Comparison between two different approximation methods.

Approximation Approach	Required Coefficients for the 16-Point DCT When $k = 7$	
	Six-Bit Dynamic Range	
	Coefficients	Error
no constraint	$-12 = -2^4 + 2^2$ $29 = 2^5 - 2^2 + 2^0$ $-25 = -2^5 + 2^3 - 2^0$ $7 = 2^3 - 2^0$	(0.0000016009) **18-bit** accuracy
$(\pm 2^p \pm 2^q)$ constraint	$30 = 2^5 - 2^1$ $9 = 2^2 + 2^0$ $-8 = -2^3$ $-3 = -2^2 + 2^0$	(0.0000102628) **16-bit** accuracy

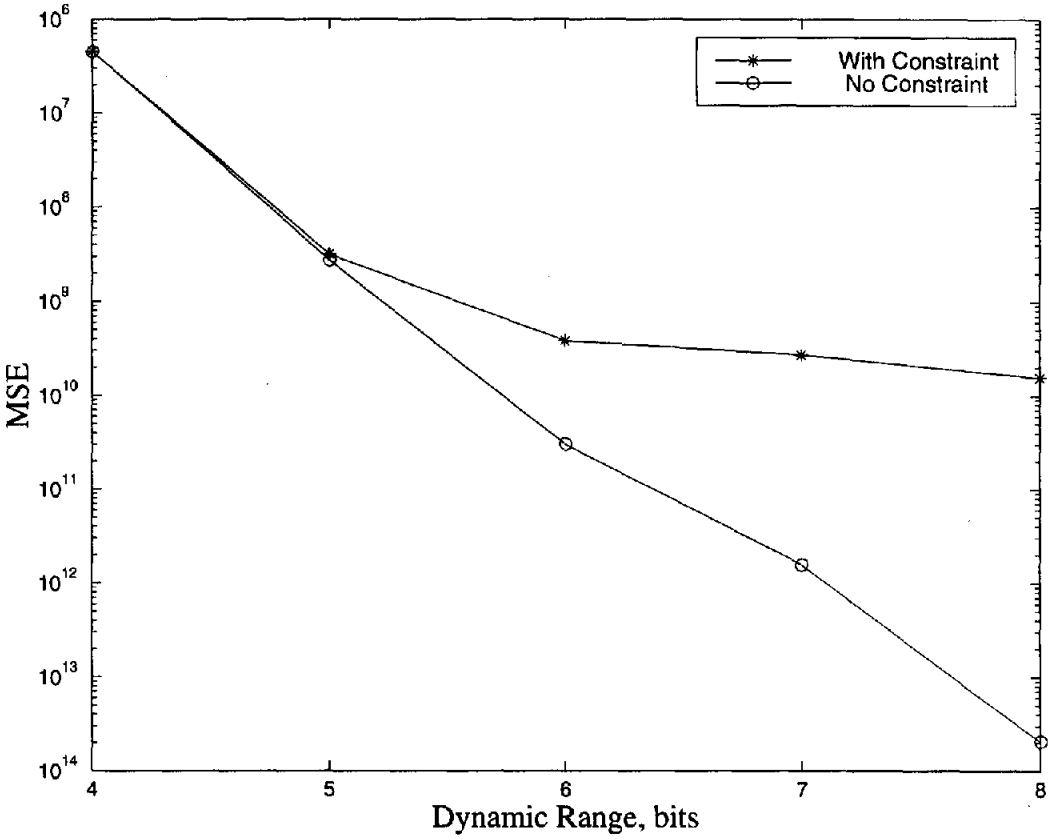

Figure 1. Error comparison of the two proposed approximation methods.

$k = 7$. As illustrated in Table 8, as a result of applying the constraint to the integers in Table 4, one multiplication is simply replaced by a maximum of two shifts and one addition.

As can be seen from Tables 6 and 7, due to the $\pm 2^p \pm 2^q$ constraint applied to the integers in Tables 4 and 5, the corresponding errors have slightly increased. In Figure 1, we have compared the mean-squared error (MSE) when computing $\cas(2\pi k/32)$ by utilizing the integers in Tables 5 and 7. In these computations, the dynamic range of the integers varies from four to eight bits (see the Appendix).

It is worth noting that the error analysis, especially in the case of multiplierless architectures, is quite important. The main theoretical hurdle is the nonuniform distribution of the algebraic

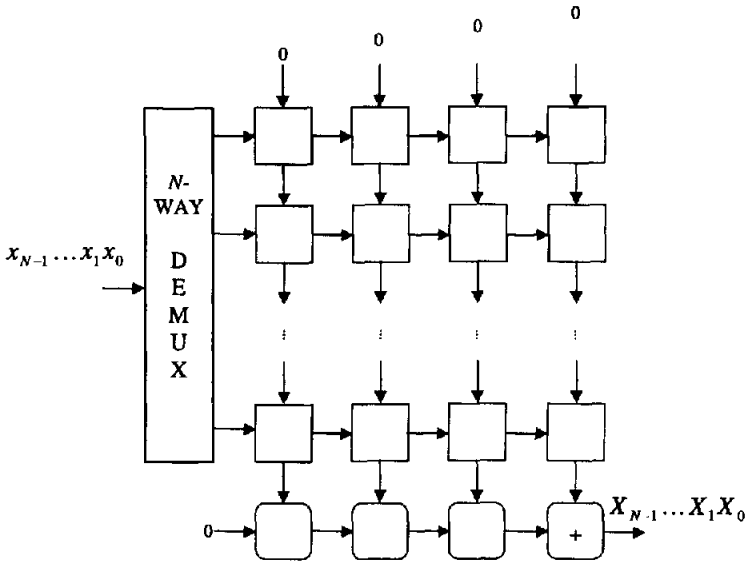

Figure 2. Systolic implementation of the N-point real-valued discrete transform when $I = 3$.

integers in the multidimensional lattice generated by them. In [16], Cozzens and Finkelstein have posed several conjectures in order to obtain a realistic picture of the error distribution and propagation in the case of the FFT. To the best of our knowledge, these conjectures are still unsolved. In real-valued transforms, we are faced with similar problems. In order to give the reader some impression about the error analysis associated with the technique proposed, we have shown several tables (see Tables 4–8 here, and Tables 11–14 in the Appendix). They realistically demonstrate the relationship between the dynamic range of the data and the approximation errors.

4. SYSTOLIC ARRAY IMPLEMENTATION

In general, there are two main approaches for the implementation of the DCT and DHT algorithms. The first method is to employ butterfly structures that lead to fast implementation of the algorithms. Although fast transformations such as fast Hartley transform (FHT) require less computation as compared to the DHT, we are faced with some common problems. Generally, the support of perfect shuffling between different stages requires global communication. This is considered a drawback when very large scale integration (VLSI) implementation of such transformation is of interest. The second method is the direct implementation that utilizes architectures such as systolic arrays [17]. The systolic implementation of transforms such as the DHT enjoys simple communication, and thus it is much more suitable for the VLSI implementation.

In this section, for the implementation of the proposed algorithms, a unified systolic architecture is presented. The proposed systolic architecture has the advantages of simplicity, modularity, and regularity [17]. Figure 2 illustrates the case where $I = 3$. In Figure 3, the cell function of each processor element (PE) is presented. This systolic array is fully pipelined, and its PEs only communicate with the neighboring cells; thus, it is very suitable for the VLSI implementation.

Let us consider the 16-point DHT that is based on the integers in Table 2. Parameter a_{ij}^t $\{i = I, \ldots, 0$ and $j, t = 0, \ldots, N-1\}$ corresponds to the integers in Table 2, and they are stored in the local registers of PE1. In a_{ij}^t, indices j and i reflect the position of PE1 in the array. Note that $j = 0$ and $i = I$ correspond to the top left PE1, where I is the degree of the polynomial of (18). Index t denotes time. In Figure 2, the input data is first pipelined into the array of PE1s through an output-buffered N-way demultiplexer. The pipelined data x_{in} should then be multiplied by integers in Table 2. However, it is easy to see that the required multiplication can

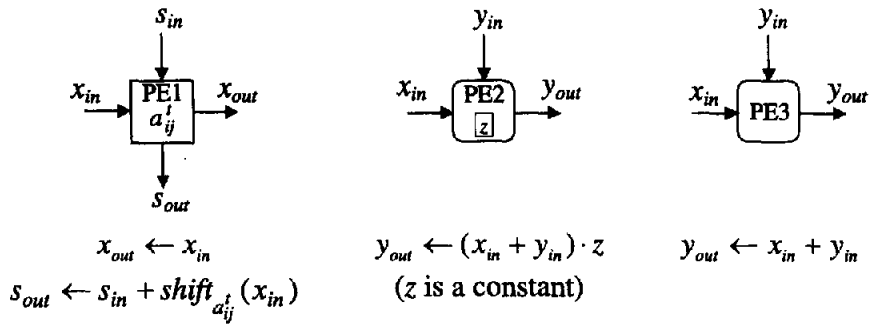

Figure 3. Input-output ports of the PEs and their cell functions.

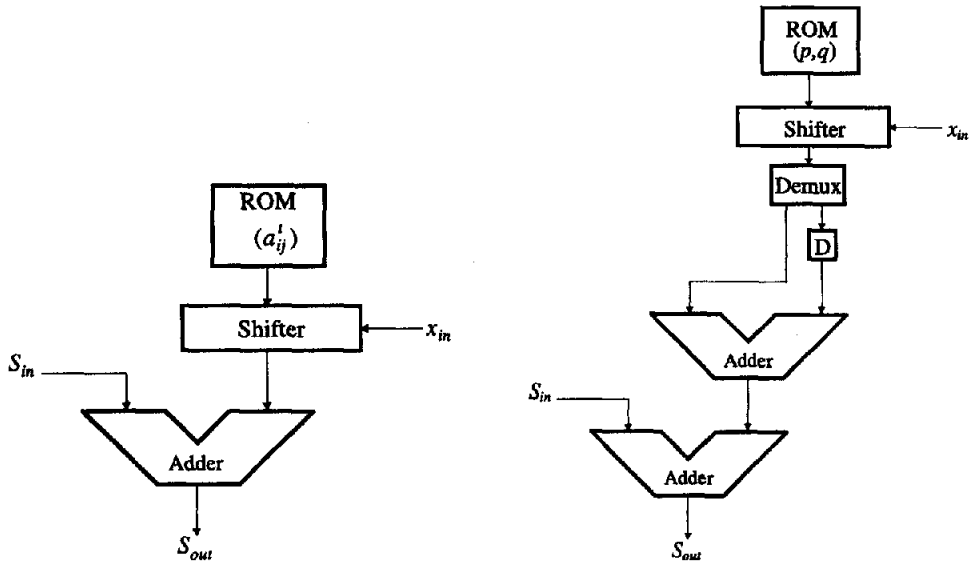

Figure 4. Different implementation techniques for $s_{\text{out}} \leftarrow s_{\text{in}} + \text{shift}_{a_{ij}^t}(x_{\text{in}})$.

be replaced with *only* one shift operation as discussed in Section 2. In PE1, the $\text{shift}_{a_{ij}^t}(x_{\text{in}})$ operation means that x_{in} is shifted once and the type of shift is determined by the a_{ij}^t coefficients that are kept in the local registers of PE1s. The shift operation can be implemented with a shifter such as a barrel shifter. Figure 4a illustrates a typical circuitry that can be utilized to implement the $s_{\text{out}} \leftarrow s_{\text{in}} + \text{shift}_{a_{ij}^t}(x_{\text{in}})$ operation. Finally, for the implementation of (19), a linear array consisting of PE2 and PE3 processors is utilized.

It is important to note that in the proposed method, we only require multiplications in PE2s. Furthermore, in these multiplications one of the multiplicands is z_2, which is constant. Thus, based on the word-length of the input data, one can estimate the precision that is needed to ensure the required accuracy. Consequently, for the required multiplications, simple special-purpose multipliers can be designed.

In the case of the DCT, the same systolic array of Figure 2 can be utilized. However, since for the N-point DCT $I = N - 1$ as compared to $N/4 - 1$ for the N-point DHT, the required number of PEs and their corresponding registers are larger. Nevertheless, by applying the approximation techniques that were presented in Section 3, hardware reductions can be achieved.

It is worth pointing out that for every order of the Haar transform, $I = 1$. As a result, the systolic architecture of Figure 2 can be utilized for the Haar transform with *only* two columns of PE1s regardless of the order of the transform.

For the implementation of the PE1, when only one shift and add operation is needed (see Table 2), the circuitry illustrated in Figure 4a can be utilized. However, for the implementation of tables in which two shifts are required, other techniques should be utilized (see Tables 6 and 7). One obvious choice is to use two shifters in PE1. This, however, may not be a practical solution in some applications. On the other hand, with a slight modification of PE1, this can be avoided. Figure 4b illustrates one possible modification. In Figure 4b, as a result of applying the constraint, the a_{ij}^t coefficients of local registers of PE1 should be replaced with the corresponding "p" and "q" coefficients. The second adder in Figure 4b can also be avoided by utilizing different techniques such as multiplexing.

5. HARDWARE AND THROUGHPUT CONSIDERATIONS

The proposed architecture requires $N(I+1)$ PE1s, I PE2s, and one adder. For evaluation of the throughput, we first assume that one time step of the global clock corresponds to one operation in PE2. For the computation of the first set of 1-D DCT/DHT, $(2N+I)$ time steps are required. The successive sets are computed in an interval of N steps. Therefore, $O(N)$ time complexity is achieved.

In our proposed method, for the calculation of the DCT/DHT, *only* I multipliers with a *constant* multiplicand are required. From Tables 1 and 2, it is obvious that for the calculation of the DCT and the DHT, 11/16 and 9/16 of the operations in PE1s are simple data transfer operations $s_{\text{out}} \leftarrow s_{\text{in}}$.

In Tables 9 and 10, different systolic implementations of the 1-D DHT and DCT are summarized and compared, respectively. As an alternative to the conventional multipliers, in some of the implementations such as [9], coordinate rotation digital computer (CORDIC) processors [18] have been employed.

Table 9. Comparison of various systolic implementations for 1-D DHT.

Length	DHT			
	Total Number of Multipliers			
	Proposed (Exact)	Proposed (Approximation)	Chang and Lee [9]	Pan and Park [3]
8	1	1	8	8
16	3	2	16	16
32	7	3	32	32
N	$(N/4-1)$	I (fixed)	N CORDICs	N

Table 10. Comparison of various systolic implementations for 1-D DCT.

Length	DCT				
	Total Number of Multipliers				
	Proposed (Exact)	Proposed (Applied to [15])	Proposed (Approximation)	Chang and Wu [8]	Pan and Park [3]
8	7	3	2	14	8
16	15	7	3	30	16
32	31	15	3	62	32
N	$(N-1)$	$(N/2-1)$	I (fixed)	$2(N-1)$	N

The main idea of our method is to use the algebraic nature of the discrete transform matrix coefficients in order to obtain more efficient algorithms. If we consider, say, DCT, then every algorithm for the DCT can be combined to the algebraic integer encoding scheme. However, it is of crucial importance to decide which one is best suited. Simulations with different algorithms show significantly different performances. Therefore, the proper choice of algorithm is essential for the successful application of the approach proposed.

6. CONCLUSIONS

In this paper, we proposed a novel approach that is aimed at efficient implementation of real-valued discrete transforms such as the DCT and the DHT. One of the main advantages of the proposed approach is an *error-free* implementation of these transformations until the final reconstruction. As a result of utilizing this approach, multiplications have been replaced by a maximum of two shift operations and one addition. The proposed method can be combined with *any* other DCT or DHT algorithms to achieve further hardware reductions. Furthermore, by introducing two approximation methods, hardware complexity was drastically reduced. Finally, for the implementation of the algorithms, a fully pipelined systolic architecture with $O(N)$ throughput was proposed.

REFERENCES

1. N. Ahmed, T. Natarajan and K.R. Rao, Discrete cosine transform, *IEEE Transactions on Computers* **23** (1), 90–93, (1974).
2. R.N. Bracewell, Discrete Hartley transform, *Journal of Optical Society of America* **73** (12), 1832–1835, (1983).
3. S.B. Pan and R.H. Park, Unified systolic arrays for computation of DCT/DST/DHT, *IEEE Transactions on Circuits and Systems for Video Technology* **7** (2), 413–419, (1997).
4. G.M. Megson, Systolic arrays for the Haar transform, *IEE Proceedings Computer and Digital Techniques* **145** (6), 403–410, (1998).
5. K.R. Rao and P. Yip, *Discrete Cosine Transform—Algorithms, Advantages, Applications*, Academic Press, New York, (1990).
6. R.N. Bracewell, Aspects of Hartley transform, *Proceedings of the IEEE* **82** (3), 381–387, (1994).
7. C. Chakrabarti and J. Jaja, Systolic architectures for the computation of the discrete Hartley and the discrete cosine transforms based on prime factor decomposition, *IEEE Transactions on Computers* **39** (11), 1359–1368, (1990).
8. L.W. Chang and M.C. Wu, A unified systolic array for discrete cosine and sine transforms, *IEEE Transactions on Signal Processing* **39** (1), 192–194, (1991).
9. L.W. Chang and S.W. Lee, Systolic arrays for the discrete Hartley transform, *IEEE Transactions on Signal Processing* **39** (11), 2411–2418, (1991).
10. J.H. Cozzens and L.A. Finkelstein, Computing the discrete Fourier transform using residue number systems in a ring of algebraic integers, *IEEE Transactions on Information Theory* **31** (5), 580–588, (1985).
11. V. Dimitrov, G.A. Jullien and W.C. Miller, A new DCT algorithm based on encoding algebraic integers, In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Vol. 3, pp. 1377–1380, Seattle, WA, (1998).
12. R. Baghaie and V. Dimitrov, DHT algorithm based on encoding algebraic integers, *IEE Electronics Letters* **35** (16), 1303–1305, (1999).
13. D.E. Knuth, *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, Addison-Wesley, (1981).
14. K.J.R. Liu, VLSI computing architectures for Haar transform, *IEE Electronics Letters* **26** (23), 1962–1963, (1990).
15. E. Feig and S. Winograd, Fast algorithms for the discrete cosine transform, *IEEE Transactions on Signal Processing* **40** (9), 2174–2193, (1992).
16. J.H. Cozzens and L.A. Finkelstein, Range and error analysis for a fast Fourier transform computed over $Z[\omega]$, *IEEE Transactions on Information Theory* **33** (4), 582–590, (1987).
17. S.Y. Kung, *VLSI Array Processors*, Prentice Hall, New Jersey, (1988).
18. J.E. Volder, The CORDIC trigonometric computing technique, *IRE Transactions on Electronic Computers* **8** (3), 330–334, (1959).

APPENDIX

In the following tables, we have included the extended version of Tables 4–7 for the seven- and eight-bit dynamic range.

Table 11. Approximation of $\cos(2\pi k/32)$ for the 16-point DCT with no constraint.

k	Seven-Bit Dynamic Range					Eight-Bit Dynamic Range				
	a_0	a_1	a_2	a_3	Error	a_0	a_1	a_2	a_3	Error
0	2	0	0	0	0	2	0	0	0	0
1	37	40	42	−40	0.0000007321	−50	−128	−128	115	0.0000000093
2	0	1	0	0	0	0	1	0	0	0
3	−51	34	48	−33	0.0000016305	91	94	56	−72	0.0000000093
4	−2	0	1	0	0	−2	0	1	0	0
5	33	−13	−6	2	0.0000015660	−109	−73	−4	41	0.0000000105
6	0	−3	0	1	0	0	−3	0	1	0
7	−12	29	−25	7	0.0000016009	27	100	−73	6	0.0000001638
8	0	0	0	0	0	0	0	0	0	0

Table 12. Approximation of $\mathrm{cas}(2\pi k/32)$ for the 32-point DHT with no constraint.

k	Seven-Bit Dynamic Range					Eight-Bit Dynamic Range				
	a_0	a_1	a_2	a_3	Error	a_0	a_1	a_2	a_3	Error
0, 8	2	0	0	0	0	2	0	0	0	0
1, 7	23	−30	−50	−15	0.0000035242	−115	48	86	−42	0.0000002312
2, 6	0	−2	0	1	0	0	−2	0	1	0
3, 5	−18	21	52	−32	0.0000000645	85	10	98	−69	0.0000000539
4	−4	0	2	0	0	−4	0	2	0	0
9, 15	47	−28	0	1	0.0000003224	47	−28	0	1	0.0000003224
10, 14	0	−4	0	1	0	0	−4	0	1	0
11, 13	−23	−30	50	−15	0.0000002730	−103	−125	122	−13	0.0000000542
12	0	0	0	0	0	0	0	0	0	0

Table 13. Approximation of $\cos(2\pi k/32)$ for the 16-point DCT with the $(\pm 2^p \pm 2^q)$ constraint.

k	Seven-Bit Dynamic Range					Eight-Bit Dynamic Range				
	a_0	a_1	a_2	a_3	Error	a_0	a_1	a_2	a_3	Error
0	2	0	0	0	0	2	0	0	0	0
1	−1	−18	18	−4	0.0000341523	96	10	4	−20	0.0000688570
2	0	1	0	0	0	0	1	0	0	0
3	−34	24	3	−3	0.0000131563	−34	24	3	−3	0.0000131563
4	−2	0	1	0	0	−2	0	1	0	0
5	56	−62	−1	10	0.0000245382	48	−62	−128	80	0.0000136974
6	0	−3	0	1	0	0	−3	0	1	0
7	30	9	−8	−3	0.0000102628	8	−72	−15	28	0.0000030869
8	0	0	0	0	0	0	0	0	0	0

Table 14. Approximation of $\mathrm{cas}(2\pi k/32)$ for the 32-point DHT with the $(\pm 2^p \pm 2^q)$ constraint.

k	Seven-Bit Dynamic Range					Eight-Bit Dynamic Range				
	a_0	a_1	a_2	a_3	Error	a_0	a_1	a_2	a_3	Error
0, 8	2	0	0	0	0	2	0	0	0	0
1, 7	−36	60	−12	−5	0.0000096490	−14	−2	65	−32	0.0000023058
2, 6	0	−2	0	1	0	0	−2	0	1	0
3, 5	−20	−18	−15	17	0.0000027199	−20	−18	−15	17	0.0000027199
4	−4	0	2	0	0	−4	0	2	0	0
9, 15	2	−60	−12	24	0.0000391315	7	−112	120	−33	0.0000316879
10, 14	0	−4	0	1	0	0	−4	0	1	0
11, 13	−2	−32	31	−7	0.0000231757	−10	−32	−96	63	0.0000150598
12	0	0	0	0	0	0	0	0	0	0